**一、获取类上的注解**

Java获取类上的注解有下面3个方法：

* Class.getAnnotations() 获取所有的注解，包括自己声明的以及继承的
* Class.getAnnotation(Class< A > annotationClass) 获取指定的注解，该注解可以是自己声明的，也可以是继承的
* Class.getDeclaredAnnotations() 获取自己声明的注解

下面，我们来演示一下3个方法的使用。   
首先，我们定义两个注解ParentAnnotation、SubAnnotation

@Retention(RetentionPolicy.RUNTIME)

@Target(value={ElementType.TYPE})

@Documented

@Inherited //可以继承

public @interface ParentAnnotation {

}

@Target(value={ElementType.TYPE})

@Retention(RetentionPolicy.RUNTIME)

@Documented

public @interface SubAnnotation {

}

接下来，我们定义两个类，Parent、Sub，分别标注ParentAnnotation 注解和SubAnnotation注解

@ParentAnnotation

public class Parent {

}

@SubAnnotation

public class Sub extends Parent{

}

一切准备OK后，就开始测试了。

public class AnnotationTest {

public static void main(String[] args) {

Annotation[] allAnnos = Sub.class.getAnnotations();

Annotation[] deAnnos = Sub.class.getDeclaredAnnotations();

Annotation subAnnotation = Sub.class.getAnnotation(SubAnnotation.class);

Annotation parentAnnotation = Sub.class.getAnnotation(ParentAnnotation.class);

printAnnotation("all",allAnnos);

printAnnotation("declare",deAnnos);

printAnnotation("sub",subAnnotation);

printAnnotation("parent",parentAnnotation);

}

private static void printAnnotation(String msg,Annotation... annotations){

System.out.println("=============="+msg+"======================");

if(annotations == null){

System.out.println("Annotation is null");

}

for (Annotation annotation : annotations) {

System.out.println(annotation);

}

System.out.println();

}

}

执行结果：

==============all======================

@com.ghs.test.annotation.ParentAnnotation()

@com.ghs.test.annotation.SubAnnotation()

==============declare======================

@com.ghs.test.annotation.SubAnnotation()

==============sub======================

@com.ghs.test.annotation.SubAnnotation()

==============parent======================

@com.ghs.test.annotation.ParentAnnotation()

尝试着将ParentAnnotation中的@Inherited去掉，结果如下：

==============all======================

@com.ghs.test.annotation.SubAnnotation()

==============declare======================

@com.ghs.test.annotation.SubAnnotation()

==============sub======================

@com.ghs.test.annotation.SubAnnotation()

==============parent======================

null

再试着将Sub类中的SubAnnotation去掉，结果如下：

==============all======================

==============declare======================

==============sub======================

null

==============parent======================

null

经过几番小小的测试，我们基本上可以得出下面几条结论：

1. 注解只有标注了@Inherited才能被子类继承
2. 当某个类没有标注任何注解时，getAnnotations()和getDeclaredAnnotations()返回空数组
3. 当某个注解查询不到时，getAnnotation(Class< A > annotationType)方法返回null

**二、获取方法上的注解**

修改上面的ParentAnnotation与SubAnnotation，使其可以标注在方法上   
@Target(value={ElementType.TYPE, ElementType.METHOD})

在Sub、Parent中分别添加一个test()方法，如下：

@ParentAnnotation

public class Parent {

@ParentAnnotation

public void test(){

}

}

@SubAnnotation

public class Sub extends Parent{

@SubAnnotation

public void test(){

}

}

一切准备就绪，就可以进行测试了。

private static void testMethodAnnotation() {

Method[] methods = Sub.class.getMethods();

for (Method method : methods) {

if(method.getName().equals("test")){

Annotation[] allMAnnos = method.getAnnotations();

Annotation[] deMAnnos = method.getDeclaredAnnotations();

Annotation subMAnno = method.getAnnotation(SubAnnotation.class);

Annotation parentMAnno = method.getAnnotation(ParentAnnotation.class);

printAnnotation("allMAnnos",allMAnnos);

printAnnotation("deMAnnos",deMAnnos);

printAnnotation("subMAnno",subMAnno);

printAnnotation("parentMAnno",parentMAnno);

}

}

}

执行结果如下：

==============allMAnnos======================

@com.ghs.test.annotation.SubAnnotation()

==============deMAnnos======================

@com.ghs.test.annotation.SubAnnotation()

==============subMAnno======================

@com.ghs.test.annotation.SubAnnotation()

==============parentMAnno======================

null

尝试着删除Sub中的test方法，再次进行测试，结果如下：

==============allMAnnos======================

@com.ghs.test.annotation.ParentAnnotation()

==============deMAnnos======================

@com.ghs.test.annotation.ParentAnnotation()

==============subMAnno======================

null

==============parentMAnno======================

@com.ghs.test.annotation.ParentAnnotation()

经过两轮测试，可以得出以下结论：

1. 子类重写的方法，注解无法被继承
2. 针对方法而言，getAnnotations()与getDeclaredAnnotations()返回的结果似乎永远都是一样的。   
   附：针对此结论，如有不同的想法，还望不吝赐教

**三、获取属性上的注解**

修改上面的ParentAnnotation与SubAnnotation，使其可以标注在属性上   
@Target(value={ElementType.TYPE, ElementType.METHOD,ElementTypeFIELD})

在Sub、Parent中分别添加一个name属性，如下：

@ParentAnnotation

public class Parent {

@ParentAnnotation

public String name;

@ParentAnnotation

public void test(){

}

}

@SubAnnotation

public class Sub extends Parent{

@SubAnnotation

public String name;

@SubAnnotation

public void test(){

}

}

下面开始测试：

private static void testFieldAnnotation() {

Field[] fields = Sub.class.getFields();

for (Field field : fields) {

Annotation[] allFAnnos= field.getAnnotations();

Annotation[] deFAnnos = field.getDeclaredAnnotations();

Annotation subFAnno = field.getAnnotation(SubAnnotation.class);

Annotation parentFAnno = field.getAnnotation(ParentAnnotation.class);

printAnnotation("allFAnnos",allFAnnos);

printAnnotation("deFAnnos",deFAnnos);

printAnnotation("subFAnno",subFAnno);

printAnnotation("parentFAnno",parentFAnno);

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}

}

执行结果如下：

==============allFAnnos======================

@com.ghs.test.annotation.SubAnnotation()

==============deFAnnos======================

@com.ghs.test.annotation.SubAnnotation()

==============subFAnno======================

@com.ghs.test.annotation.SubAnnotation()

==============parentFAnno======================

null

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

==============allFAnnos======================

@com.ghs.test.annotation.ParentAnnotation()

==============deFAnnos======================

@com.ghs.test.annotation.ParentAnnotation()

==============subFAnno======================

null

==============parentFAnno======================

@com.ghs.test.annotation.ParentAnnotation()

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

经过测试，我们可以得出下面的几个结论：

1. 父类的属性和子类的属性互补干涉
2. 针对属性而言，getAnnotations()与getDeclaredAnnotations()方法返回的结果似乎都是一样的   
   附：针对此结论，如有不同的想法，还望不吝赐教